Изображение выглядит как цепь, медальон

Автоматически созданное описание

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.03.01 | Информатика и вычислительная техника | | |
| Направленность (профиль) | |  | Системы автоматизированного проектирования | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Программирование | | |
| Курс | I | | | Группа | 414 |

Отчёт по контрольной работе № 2

Вариант № 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 414 |  |  |  | Кондиляброва  Вероника Данииловна |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверили: |  |  |  | Корниенко Иван Григорьевич |
|  |  | (дата, подпись) |  | Федин Алексей Константинович |

Содержание

[1 Постановка задачи 3](#_Toc114812418)

[2 Исходные данные 3](#_Toc114812419)

[3 Особые ситуации 3](#_Toc114812420)

[4 Математические методы и алгоритмы решения задач 3](#_Toc114812421)

[5 Форматы представления данных 4](#_Toc114812422)

[6 Структура программы 5](#_Toc114812423)

[7 Описание хода выполнения 6](#_Toc114812424)

[8 Блок-схема алгоритма решения задачи 7](#_Toc114812425)

[9 Результаты работы программы 9](#_Toc114812426)

[10 Исходный код полученного программного решения 14](#_Toc114812427)

## 1 Постановка задачи

В контрольной работе задается базовый и производный класс. Необходимо разработать поля и методы, наследуемые из базового класса, а также собственные компоненты производных классов. Базовый класс может быть абстрактным. Реализовать возможность получения списка объектов в контейнере.

Базовый класс – товар; поля – наименование, количество, производитель, цена за единицу изделия, общая цена. Производный класс – товар на складе.

## 2 Исходные данные

В качестве исходных данных программа использует количество продуктов и их описание: название, количество данного продукта, производителя, цену за единицу товара, общую стоимость и наличие продукта на складе для составления и сортировки списка. Ввод с клавиатуры или из файла.

## 3 Особые ситуации

1 Пользователь может ввести число абитуриентов меньше одного, устранено с помощью функции GetInt, осуществляющей проверку ввода;

2 Пользователь может ввести вместо названия и производителя продукта цифры, устранено с помощью функции GetString, осуществляющей проверку ввода на символы;

3 Пользователь может неправильно указать наличие продукта, устранено с помощью функции GetAbailability(), которая позволяет ввести только 4 варианта “Have”, “have”, “Nothave” или “nothave”

4 Отсутствие ожидаемых программой файлов устранено методом open. Если программа не находит файл с введенным пользователем именем, этот метод сам его создает.

## 4 Математические методы и алгоритмы решения задач

Алгоритм для сортировки

Сортировка по наличию продукта:

1. Прохождение по всему массиву;
2. Проверка на наличие продукта на складе;
3. Если продукт есть на складе, он добавляется в новый массив;

Алгоритм работы пользовательского интерфейса

1. Цикл while, отвечающий за выход из программы;
2. Оператор множественного выбора switch – case осуществляет остальные действия в работе с пользователем

## 5 Форматы представления данных

Таблица 1 – Переменные

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| instock | CInStock\* | Массив для каждого объекта |
| product\_in\_stock | vector <class CInStock> | Массив объектов |
| user\_choice | int | Пользовательский выбор в главном меню |
| exit | int | Выход из меню |
| numb | int | Пользовательский ввод количества клиентов |

Базовый класс CProduct

Таблица 2 – Поля

|  |  |  |
| --- | --- | --- |
| Имя | Модификатор | Тип |
| Name | private | string |
| Volume | private | int |
| Maker | private | string |
| Price | private | int |
| Total\_price | private | int |

Таблица 3 – Методы

|  |  |  |
| --- | --- | --- |
| Имя | Модификатор | Описание |
| set | public | Ввод данных продукта |
| get | public | Передает каждое поле |
| show | public | Вывод данных продукта |

Таблица 4 – Конструкторы:

|  |  |  |
| --- | --- | --- |
| Имя | Принимаемые значения | Описание |
| CProduct | - | По умолчанию |
| CProduct | string Name, int Volume, string Maker, int Price, int Total\_price | Копирования |

Производный класс CInStock

Таблица 5 – Поля

|  |  |  |
| --- | --- | --- |
| Имя | Модификатор | Тип |
| Availability | private | string |

Таблица 6 – Методы

|  |  |  |
| --- | --- | --- |
| Имя | Модификатор | Описание |
| set | public | Ввод данных продукта |
| get | public | Передает каждое поле |
| show | public | Вывод данных продукта |

Таблица 7 – Конструкторы:

|  |  |  |
| --- | --- | --- |
| Имя | Принимаемые значения | Описание |
| CInStock | - | По умолчанию |
| CInStock | string Name, int Volume, string Maker, int Price, int Total\_price | Копирования |

## 6 Структура программы

Таблица 8 – Структура программы

|  |  |  |
| --- | --- | --- |
| Модуль | Функция | Описание |
| Correct\_input | GetString | Проверка ввода букв |
|  | GetInt | Проверка ввода количества клиентов и в общем чисел |
|  | GetAvailability | Проверка ввода наличия продукта на складе |
| Function | CreateStock | Создаёт массив продуктов |
|  | InFile | Заполняет массив продуктов из файла |
|  | Availability | Показывают продукты, находящиеся на складе |
|  | Delete\_product | Удаляет выбранные продукты |
|  | LookProducts | Показывает все продукты |
|  | SaveInFile | Сохраняет в файл |
| Menu | MenuMain | Осуществляет работу пользовательского интерфейса |
| Main | main | Вызывает функцию MenuMain, с которой начинается выполнение программы |
| Test | test | Тестирует работу программы |

## 7 Описание хода выполнения

При выполнении контрольной работы я:

* Освоила работу с итераторами STL и контейнером vector;
* Изучила наследование классов;

Решенные возникшие проблемы:

* Считать данные из файла – с помощью объекта ifstream;
* Обозначение поля наличие продукта на складе для пользователя;
* Сортировка по наличию продукта на складе (пыталась удалять неподходящие элементы) – добавила в пустой контейнер подходящие;

Непонятные моменты:

* Проходить по контейнеру с помощью итератора циклом for;

## 8 Блок-схема алгоритма решения задачи

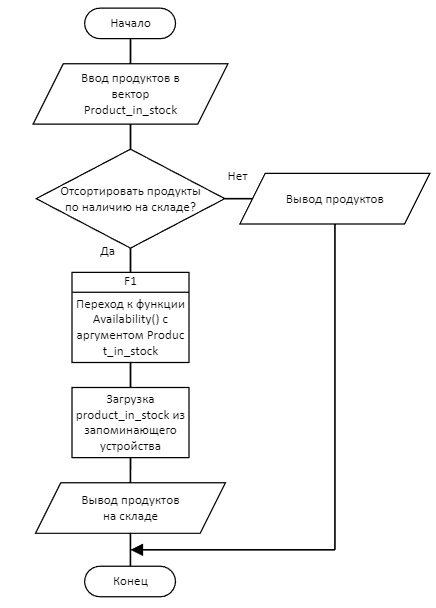


Рисунок 1 – Блок-схема алгоритма решения

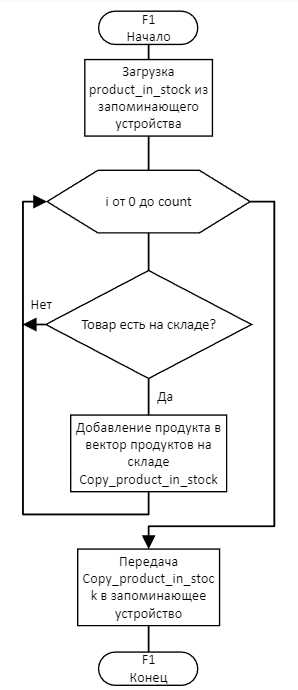


Рисунок 2 – Блок-схема алгоритма сортировки по наличию на складе

## 9 Результаты работы программы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Тест

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Проверка на ввод цифр, букв и пункта «наличие продукта»

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 – Фильтр (только склад)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 – Просмотреть все продукты

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – Удаление продуктов

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 – Продукты из файла

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 – Cохранение в файл

## 10 Исходный код полученного программного решения

//Menu.h

#pragma once

#include<iostream>

#include <vector>

#include"ChildClass.h"

#include"CorrectInput.h"

#include"Test.h"

enum {

Test=1,

Input,

Delete,

Look\_products,

Available\_in\_stock,

Exit,

};

enum {

Add\_manual=1,

Add\_file,

Back,

};

void MainMenu();

vector<class CInStock> InputMenu();

//Menu.cpp

#include "Menu.h"

#include"Function.h"

using namespace std;

void MainMenu() {

bool check\_enter;

vector<class CInStock>product\_in\_stock;

vector<class CInStock> Copy\_product\_in\_stock;

bool exit = 0;

while (exit == false) {

cout << "Test - 1\nEnter data - 2\nDelete products - 3\nLook products - 4\nLook only products in stock - 5\nExit - 6\nChoose: ";

int out\_user\_choice = GetInt();

switch (out\_user\_choice) {

case(Test): {

TestProducts();

}break;

case(Input):{

product\_in\_stock=InputMenu();

}break;

case(Delete): {

check\_enter = CheckExistanceData(product\_in\_stock);

if (check\_enter == true) {

DeleteProduct(product\_in\_stock);

cout << "Do you want to save? Yes - 1, No - 2: ";

int save\_choice = GetSaveChoice();

if (save\_choice == 1) {

SaveInFile(product\_in\_stock);

}

if (save\_choice == 2) {

break;

}

}

else {

cout << "You don't enter data. Please, choose item '2'" << endl;

break;

}

}break;

case(Look\_products): {

check\_enter = CheckExistanceData(product\_in\_stock);

if (check\_enter == true) {

LookProducts(product\_in\_stock);

cout << "Do you want to save? Yes - 1, No - 2: ";

int save\_choice = GetSaveChoice();

if (save\_choice == 1) {

SaveInFile(product\_in\_stock);

}

if (save\_choice == 2) {

break;

}

}

else {

cout << "You don't enter data. Please, choose item '2'" << endl;

break;

}

}break;

case(Available\_in\_stock): {

check\_enter = CheckExistanceData(product\_in\_stock);

if (check\_enter == true) {

Copy\_product\_in\_stock = Availability(product\_in\_stock);

if ( static\_cast<int>(Copy\_product\_in\_stock.size()) == 0) {

cout << endl << "No products" << endl << endl;

}

else {

for (int i = 0; i< static\_cast<int>(Copy\_product\_in\_stock.size()); i++) {

Copy\_product\_in\_stock[i].Show(i);

}

cout << "Do you want to save? Yes - 1, No - 2: ";

int save\_choice = GetSaveChoice();

if (save\_choice == 1) {

SaveInFile(Copy\_product\_in\_stock);

}

if (save\_choice == 2) {

break;

}

}

}

else {

cout << "You don't enter data. Please, choose item '2'" << endl;

break;

}

}break;

case(Exit): {

exit = true;

}break;

default: {

cout << "Error. Try again." << endl;

}break;

}

}

}

vector<class CInStock> InputMenu() {

bool exit = false;

vector<class CInStock>Product\_in\_stock;

while (exit == false) {

cout << "Add products manual - 1\nAdd of file - 2\nBack - 3\nChoose: ";

int in\_user\_choice = GetInt();

switch (in\_user\_choice) {

case(Add\_manual): {

CreateStock(Product\_in\_stock);

cout << "Do you want to save? Yes - 1, No - 2: ";

int save\_choice = GetSaveChoice();

if (save\_choice == 1) {

SaveInFile(Product\_in\_stock);

}

if (save\_choice == 2) {

break;

}

}break;

case(Add\_file): {

InFile(Product\_in\_stock);

}break;

case(Back): {

exit = true;

}break;

default: {

cout << "Error. Try again." << endl;

}break;

}

}

return Product\_in\_stock;

}

//ChildClass.h

#pragma once

#include "ParentClass.h"

class CInStock : public CProduct {

private:

string Availability;

public:

void SetAvailability(string Availability);

string GetAvailability();

void Show(int i);

CInStock(string Name, int Volume, string Maker, int Price, int Total\_price, string Availability);

CInStock();

~CInStock();

};

//ChildClass.cpp

#include <iostream>

#include <string>

#include "ChildClass.h"

using namespace std;

void CInStock::SetAvailability(string availability) {

this->Availability = availability;

}

string CInStock::GetAvailability() {

return Availability;

}

void CInStock::Show(int i) {

cout << "Product " << i + 1 << ":\n" << GetName() << " " << GetVolume() << " pieces\n" << "Maker: " << GetMaker() << "\n" << "1 pieces cost " << GetPrice() //

<< "\n" << "Total price: " << GetTotal\_price() << "\n" << "Product " << GetAvailability() << endl;

}

CInStock::CInStock(string Name, int Volume, string Maker, int Price, int Total\_price, string Availability) {

this->Name = Name;

this->Volume = Volume;

this->Maker = Maker;

this->Price = Price;

this->Total\_price = Total\_price;

this->Availability = Availability;

}

CInStock::CInStock() {

CProduct();

}

CInStock::~CInStock() {};

//ParentClass.h

#pragma once

#include <string>

using namespace std;

class CProduct {

protected:

string Name;

int Volume;

string Maker;

int Price;

int Total\_price;

public:

void SetName(string Name);

string GetName();

void SetVolume(int Volume);

int GetVolume();

void SetMaker(string Maker);

string GetMaker();

void SetPrice(int Price);

int GetPrice();

void SetTotal\_price(int Total\_price);

int GetTotal\_price();

CProduct(string Name, int Volume, string Maker, int Price, int Total\_price);

CProduct();

virtual ~CProduct();

};

//ParentClass.cpp

#include <iostream>

#include <string>

#include "ParentClass.h"

using namespace std;

void CProduct::SetName(string name) {

this->Name = name;

}

string CProduct::GetName() {

return Name;

}

void CProduct::SetVolume(int volume) {

this->Volume = volume;

}

int CProduct::GetVolume() {

return Volume;

}

void CProduct::SetMaker(string maker) {

this->Maker = maker;

}

string CProduct::GetMaker() {

return Maker;

}

void CProduct::SetPrice(int price) {

this->Price = price;

}

int CProduct::GetPrice() {

return Price;

}

void CProduct::SetTotal\_price(int total\_price) {

this->Total\_price = total\_price;

}

int CProduct::GetTotal\_price() {

return Total\_price;

}

CProduct::CProduct(string Name, int Volume, string Maker, int Price, int Total\_price) {

this->Name = Name;

this->Volume = Volume;

this->Maker = Maker;

this->Price = Price;

this->Total\_price = Total\_price;

}

CProduct::CProduct() {

}

CProduct::~CProduct() {}

//CorrectInput.h

#pragma once

#include<string>

using namespace std;

string GetString();

int GetInt();

string GetAbailability();

int GetSaveChoice();

bool GetBool();

//CorrectInput.cpp

#include <iostream>

#include "CorrectInput.h"

using namespace std;

string GetString() {

int count = 0;

int flag = 0;

string Enter;

while (flag != 1) {

cin >> Enter;

char Symbol;

int len = (int)Enter.length();

for (int i = 0; i < len; i++) {

Symbol = Enter[i];

if (!((Symbol >= 'A' && Symbol <= 'Z') || (Symbol >= 'a' && Symbol <= 'z'))) {

count += 1;

}

}

if (count > 0) {

cout << "Error! Try again." << endl;

cin.clear();

cin.ignore(INT\_MAX, '\n');

count = 0;

}

else {

flag = 1;

}

}

return Enter;

}

int GetInt() {

int input;

cin >> input;

while (!(input >= 1)) {

cout << "Error! Try again." << endl;

cin.clear();

cin.sync();

cin.ignore(INT\_MAX, '\n');

cin >> input;

}

return input;

}

string GetAbailability() {

int flag = 0;

string Enter;

while (flag != 1) {

cin >> Enter;

if (!(Enter=="Have" || Enter=="have" || Enter=="nothave" || Enter=="Nothave")) {

cout << "Error! Try again." << endl;

cin.clear();

cin.ignore(INT\_MAX, '\n');

}

else {

flag = 1;

}

}

return Enter;

}

int GetSaveChoice() {

int input;

cin >> input;

while (!(input >= 1 && input <= 2)) {

cout << "Error. Try again." << endl;

cin.clear();

cin.sync();

cin.ignore(INT\_MAX, '\n');

cin >> input;

}

return input;

}

bool GetBool() {

bool input;

cin >> input;

while (!(input == 0 || input == 1)) {

cout << "Error. Try again." << endl;

cin.clear();

cin.sync();

cin.ignore(INT\_MAX, '\n');

cin >> input;

}

return input;

}

//Function.h

#pragma once

#include<iostream>

#include<fstream>

#include"ChildClass.h"

#include"CorrectInput.h"

#include "Menu.h"

void CreateStock(vector<class CInStock>&product\_in\_stock);

void InFile(vector<class CInStock>& product\_in\_stock);

vector<class CInStock> Availability(vector<class CInStock>& product\_in\_stock);

vector<class CInStock>& DeleteProduct(vector<class CInStock>& product\_in\_stock);

void LookProducts(vector<class CInStock>& product\_in\_stock);

void SaveInFile(vector<class CInStock>& product\_in\_stock);

bool CheckExistanceData(vector<class CInStock>& product\_in\_stock);

//Function.cpp

#include"Function.h"

void CreateStock(vector<class CInStock>& product\_in\_stock) {

string Str = "";

int Int = 0;

cout << "Enter number of products: ";

int numb = GetInt();

CInStock\* instock = new CInStock[numb];

for (int i = 0; i < numb; i++) {

cout << "Name of product: ";

Str = GetString();

instock[i].SetName(Str);

cout << "Volume of product: ";

Int = GetInt();

instock[i].SetVolume(Int);

cout << "Maker of product: ";

Str = GetString();

instock[i].SetMaker(Str);

cout << "Price of product: ";

Int = GetInt();

instock[i].SetPrice(Int);

cout << "Total price of product: ";

Int = GetInt();

instock[i].SetTotal\_price(Int);

cout << "Available on stock (have or nothave): ";

Str = GetAbailability();

instock[i].SetAvailability(Str);

product\_in\_stock.push\_back(instock[i]);

}

delete[] instock;

instock = nullptr;

}

void InFile(vector<class CInStock>& product\_in\_stock) {

cout << "Enter the name of file(without file permission!): ";

string name\_file = "";

cin >> name\_file;

ifstream file;

file.open(name\_file);

if (!file.is\_open()) {

cout << "File opening error" << endl;

}

else {

while (!file.eof()) {

string name = "";

int volume = {};

string maker = "";

int price = {};

int total\_price = {};

string availability = "";

file >> name;

file >> volume;

file >> maker;

file>> price;

file>> total\_price;

file >> availability;

CInStock Product\_in\_stock(name, volume, maker, price, total\_price, availability);

product\_in\_stock.push\_back(Product\_in\_stock);

}

}

file.close();

}

vector<class CInStock>& DeleteProduct(vector<class CInStock>& product\_in\_stock) {

vector<class CInStock>::iterator iter = product\_in\_stock.begin();

cout << "Enter number of product you want to delete: ";

int number\_for\_delete = GetInt();

iter += (number\_for\_delete - 1);

product\_in\_stock.erase(iter);

return product\_in\_stock;

}

void LookProducts(vector<class CInStock>& product\_in\_stock) {

for (int i = 0; i < static\_cast<int>(product\_in\_stock.size()); i++) {

product\_in\_stock[i].Show(i);

}

}

vector<class CInStock> Availability(vector<class CInStock>& product\_in\_stock) {

vector<class CInStock>Copy\_product\_in\_stock;

for (int i = 0; i < static\_cast<int>(product\_in\_stock.size()); i++) {

if (product\_in\_stock[i].GetAvailability() == "Have" || product\_in\_stock[i].GetAvailability() == "have") {

Copy\_product\_in\_stock.push\_back(product\_in\_stock[i]);

}

}

return Copy\_product\_in\_stock;

}

void SaveInFile(vector<class CInStock>& product\_in\_stock) {

int tryAnotherFile{};

ofstream file;

string name\_file = "";

cout << "Enter the name of file(without file permission!): ";

cin >> name\_file;

if (ifstream(name\_file)) {

cout << "The file exists." << endl;

cout << "Overwrite existing file - 0\nEnter again - 1\nChoose: ";

tryAnotherFile = GetBool();

}

file.open(name\_file); // ф-ция open, если не находит файл с таким именем.

if (!file.is\_open()) {

cout << "File opening error";

}

else {

if (tryAnotherFile == 0) {

for (int i = 0; i < static\_cast<int>(product\_in\_stock.size()); i++) {

file << "Product " << i + 1 << ":\n" << product\_in\_stock[i].GetName() << " " << product\_in\_stock[i].GetVolume() << " pieces\n" << "Maker: " //

<< product\_in\_stock[i].GetMaker() << "\n" << "1 pieces cost " << product\_in\_stock[i].GetPrice() << "\n" << "Total price: " << product\_in\_stock[i].GetTotal\_price() << //

"\n" << "Product " << product\_in\_stock[i].GetAvailability() << endl;

}

}

if (tryAnotherFile == 1) {

SaveInFile(product\_in\_stock);

}

}

file.close();

}

bool CheckExistanceData(vector<class CInStock>& product\_in\_stock) {

bool flag = true;

if ( static\_cast<int>(product\_in\_stock.size()!=0)) {

for (int i = 0; i < static\_cast<int>(product\_in\_stock.size()); i++) {

if (product\_in\_stock[i].GetName() == "" && product\_in\_stock[i].GetMaker() == "" && product\_in\_stock[i].GetVolume() == 0 && product\_in\_stock[i].GetPrice() == 0 && product\_in\_stock[i].GetTotal\_price() == 0 && product\_in\_stock[i].GetAvailability() == "") {

flag = false;

}

}

}

else {

flag = false;

}

return flag;

}

//Test.h

#pragma once

#include "Function.h"

#include<iostream>

using namespace std;

void TestProducts();

//Test.cpp

#include"Test.h"

void TestProducts() {

bool Check\_test1 = false;

bool Check\_test2 = false;

bool Check\_test3 = false;

vector<class CInStock> Test1;

vector<class CInStock> Copy\_test1;

vector<class CInStock> Test\_result1;

vector<class CInStock> Test2;

vector<class CInStock> Copy\_test2;

vector<class CInStock> Test\_result2;

vector<class CInStock> Test3;

vector<class CInStock> Copy\_test3;

CInStock Product1("Apple", 5, "LoveApple", 10, 50, "have");

CInStock Product2("Orange", 7, "LoveOrange", 12, 84, "nothave");

CInStock Product3("Mango", 2, "LoveMango", 50, 100, "Have");

Test1.push\_back(Product1);

Test1.push\_back(Product2);

Test1.push\_back(Product3);

Test\_result1.push\_back(Product1);

Test\_result1.push\_back(Product3);

Copy\_test1 = Availability(Test1);

CInStock Product4("Apple", 5, "LoveApple", 10, 50, "Nothave");

CInStock Product5("Orange", 7, "LoveOrange", 12, 84, "nothave");

CInStock Product6("Mango", 2, "LoveMango", 50, 100, "have");

Test2.push\_back(Product4);

Test2.push\_back(Product5);

Test2.push\_back(Product6);

Test\_result2.push\_back(Product6);

Copy\_test2=Availability(Test2);

CInStock Product7("Apple", 5, "LoveApple", 10, 50, "Nothave");

CInStock Product8("Orange", 7, "LoveOrange", 12, 84, "nothave");

CInStock Product9("Mango", 2, "LoveMango", 50, 100, "Nothave");

Test3.push\_back(Product7);

Test3.push\_back(Product8);

Test3.push\_back(Product9);

Copy\_test3 = Availability(Test3);

if ( static\_cast<int>(Copy\_test1.size()) == static\_cast<int>(Test\_result1.size())) {

for (int i = 0; i < static\_cast<int>(Copy\_test1.size()); i++) {

if ((Copy\_test1[i].GetAvailability() == Test\_result1[i].GetAvailability() && Copy\_test1[i].GetMaker() == Test\_result1[i].GetMaker()

&& Copy\_test1[i].GetName() == Test\_result1[i].GetName() && Copy\_test1[i].GetPrice() == Test\_result1[i].GetPrice()

&& Copy\_test1[i].GetTotal\_price() == Test\_result1[i].GetTotal\_price() && Copy\_test1[i].GetVolume() == Test\_result1[i].GetVolume())) {

Check\_test1 = true;

}

else {

Check\_test1 = false;

cout << endl << "Test №1 was negative." << endl << endl;

cout << "The result of the work:" << endl;

for (int j = 0; j < static\_cast<int>(Copy\_test1.size()); j++) {

Copy\_test1[i].Show(j);

}

cout << endl << "Expected result:" << endl;

for (int k = 0; k < static\_cast<int>(Test\_result1.size()); k++) {

Test\_result1[i].Show(k);

}

}

}

}

else {

Check\_test1 = false;

cout << endl << "Test №1 was negative." << endl << endl;

cout << "The result of the work:" << endl;

for (int i = 0; i < static\_cast<int>(Copy\_test1.size()); i++) {

Copy\_test1[i].Show(i);

}

cout << endl << "Expected result:" << endl;

for (int i = 0; i < static\_cast<int>(Test\_result1.size()); i++) {

Test\_result1[i].Show(i);

}

}

if ( static\_cast<int>(Copy\_test2.size()) == static\_cast<int>(Test\_result2.size())) {

for (int n = 0; n < static\_cast<int>(Copy\_test2.size()); n++) {

if (Copy\_test2[n].GetAvailability() == Test\_result2[n].GetAvailability() && Copy\_test2[n].GetMaker() == Test\_result2[n].GetMaker()

&& Copy\_test2[n].GetName() == Test\_result2[n].GetName() && Copy\_test2[n].GetPrice() == Test\_result2[n].GetPrice()

&& Copy\_test2[n].GetTotal\_price() == Test\_result2[n].GetTotal\_price() && Copy\_test2[n].GetVolume() == Test\_result2[n].GetVolume()) {

Check\_test2 = true;

}

else {

Check\_test2 = false;

cout << endl << "Test №2 was negative." << endl << endl;

cout << "The result of the work:" << endl;

for (int m = 0; m < static\_cast<int>(Copy\_test2.size()); m++) {

Copy\_test2[m].Show(m);

}

cout << endl << "Expected result:" << endl;

for (int f = 0; f < static\_cast<int>(Test\_result2.size()); f++) {

Test\_result2[f].Show(f);

}

}

}

}

else {

Check\_test2 = false;

cout << endl << "Test №2 was negative." << endl << endl;

cout << "The result of the work:" << endl;

for (int i = 0; i < static\_cast<int>(Copy\_test2.size()); i++) {

Copy\_test2[i].Show(i);

}

cout << endl << "Expected result:" << endl;

for (int i = 0; i < static\_cast<int>(Test\_result2.size()); i++) {

Test\_result2[i].Show(i);

}

}

if ( static\_cast<int>(Copy\_test3.size()) == 0) {

Check\_test3 = true;

}

else {

Check\_test3 = false;

cout << endl << "Test №3 was negative." << endl << endl;

cout << "The result of the work:" << endl;

for (int i = 0; i < static\_cast<int>(Copy\_test3.size()); i++) {

Copy\_test3[i].Show(i);

}

cout << endl << "Expected result:" << endl;

cout << "No products" << endl;

}

if (Check\_test1 == true && Check\_test2 == true && Check\_test3 == true) {

cout << endl << "Test was successful" << endl;

}

}

//Main.cpp

#include <iostream>

#include<string>

#include <vector>

#include"Menu.h"

using namespace std;

int main() {

setlocale(LC\_ALL, "ru");

cout << "Kondilyabrova Veronika Daniilovna\nGroup 414" << endl;

cout << "Control work №2\nVariant 10\nCreate a base class – product; fields – name, quantity, manufacturer, unit price, total price.\nDerived class – products in stock.\n\n" << endl;

MainMenu();

return 0;

}